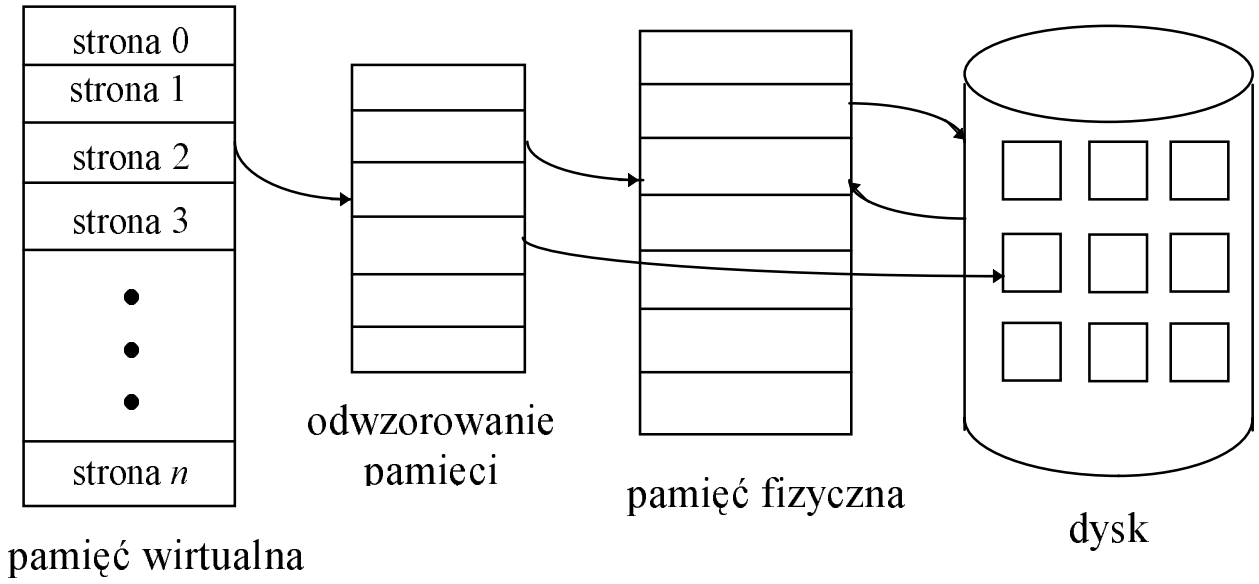
Pamięć wirtualna

- Umożliwia wykonywanie procesów, pomimo że nie są one w całości przechowywane w pamięci operacyjnej
- Logiczna przestrzeń adresowa może być dużo większa od fizycznej przestrzeni adresowej
- Tworzy iluzję dużej, jednorodnej i szybkiej pamięci
- Techniki realizacji pamięci wirtualnej: stronicowanie na żądanie, segmentacja na żądanie, segmentacja ze stronicowaniem



Stronicowanie na żądanie

- Strona jest sprowadzana do pamięci dopiero wtedy, gdy jest potrzebna:
  mniej wejścia-wyjścia, mniejsze zapotrzebowanie na pamięć, więcej użytkowników
- Strona jest potrzebna, gdy pojawia się do niej odwołanie
- Z każdą pozycją w tablicy stron jest związany *bit poprawności odwołania* (1 - strona w pamięci, 0 - strona poza pamięcią)

- Odwołanie do strony z bitem poprawności odwołania równym 0 powoduje ***błąd braku strony*** (ang. *page fault*). System operacyjny:
  - znajduje wolną ramkę
  - sprowadza stronę z dysku do pamięci
  - uaktualnia bit poprawności odwołania (:= 1)
  - restartuje instrukcję, która spowodowała błąd
- Co się dzieje, gdy nie ma wolnej ramki:
  wymiana stron (ang. *page replacement*) - SO znajduje w pamięci stronę (najmniej używaną) i usuwa ją na dysk
- Niektóre strony mogą być sprowadzane do pamięci wielokrotnie

Wydajność stronicowania na żądanie:

p = współczynnik błędów braku strony $0 \le p \le 1.0$

$p = 0 \Rightarrow$ nie ma błędów braku strony

$p = 1 \Rightarrow$ każde odwołanie powoduje błąd braku strony

efektywny czas dostępu =

$(1 - p) \times$ *czas dostępu do pamięci*

$+ p \times$ *czas obsługi błędu braku strony*

obsługa błędu braku strony = czas obsługi przerwania + [czas wysłania strony na dysk] + czas sprowadzenia strony z dysku + czas wznowienia obliczeń

Algorytmy wymiany stron:

- *Bit modyfikacji* umożliwia zmniejszenie liczby transmisji dyskowych - tylko modyfikowane strony trzeba zapisywać z powrotem na dysk
- Algorytm powinien minimalizować liczbę błędów strony

- Ciąg odwołań: ciąg numerów stron, np.
  1, 2, 3, 4, 1, 2, 5, 1, 2, 3, 4, 5
- **Algorytm FIFO**
  3 ramki pamięci ⇒ 9 przerwań braku strony
  4 ramki pamięci ⇒ 10 przerwań braku strony
  (*anomalia FIFO*; FIFO nie jest *algorytmem stosowym*)
- **Algorytm optymalny**: usuń stronę, do której najdłużej nie będzie odwołania (alg. stosowy: $M(m,r) \subseteq M(m+1,r)$)
  3 ramki pamięci ⇒ 7 przerwań braku strony
  4 ramki pamięci ⇒ 6 przerwań braku strony
  Nierealizowalny w praktyce, stosowany do porównań
- **Algorytm LRU** („najdłużej nieużywany najpierw")
  3 ramki pamięci ⇒ 10 przerwań braku strony
  4 ramki pamięci ⇒ 8 przerwań braku strony
  Implementacja z licznikiem: z każdą pozycją w tablicy stron jest związany licznik, podczas każdego odwołania zawartość zegara kopiuje się do licznika
  Implementacja ze stosem: numery stron przechowuje się na stosie, po każdym odwołaniu do strony przesuwa się jej numer na wierzchołek stosu
  Algorytmy przybliżające LRU:
  1. *Bit odwołania*: z każdą stroną związuje się bit odwołania (inicjalnie = 0); w chwili odwołania do strony ustawia się bit na 1; usuwa się stronę z bitem 0; którą?
  2. *Algorytm drugiej szansy*: przegląda się strony zgodnie z porządkiem FIFO, cyklicznie; gdy bit=0, to stronę można wymienić; gdy bit=1; to zeruje się bit, a stronie daje „drugą szansę" (czas sprow. do pamięci:=czas bieżący)
  3. *Algorytm zegarowy*: jak w 2, lecz cykliczna lista stron
- **Algorytm LFU** („najmniej używana")
- **Algorytm MFU** („najwięcej używana")

Przydział ramek:

- Minimalna liczba ramek zależy od architektury komputera
- *Przydział równy*: każdy proces dostaje tyle samo
- *Przydział proporcjonalny*: każdy proces dostaje proporcjonalnie do swojego rozmiaru
- *Przydział priorytetowy*:proces o wyższym priorytecie może wybrać stronę do usunięcia spośród swoich stron lub stron procesów o niższym priorytecie

Algorytmy lokalne (wybór strony do usunięcia spośród własnych stron) i **globalne** (wybór strony do usunięcia spośród wszystkich stron)

Migotanie

Proces jest zajęty głównie przesyłaniem stron z dysku do pamięci i z pamięci na dysk
Brak wystarczającej liczby stron jest powodem wysokiego współczynnika błędów braku strony:
⇒ niskie wykorzystanie CPU
⇒ system operacyjny myśli, że trzeba zwiększyć poziom wieloprogramowości
⇒ do systemu dodaje się nowy proces
Wykres zależności wykorzystania procesora od poziomu wieloprogramowości

Model pola roboczego (ang. *working-set*)

Zasada lokalności odwołań: w każdej fazie wykonania program korzysta jedynie z drobnej części całego zbioru stron (lokalność *czasowa*, *przestrzenna*)
T = parametr ⇒ *okienko* (ustalona liczba odwołań do stron)

Pole robocze procesu P_i (WS_i): całkowita liczba stron, do których proces odwołał się podczas ostatnich T odwołań

Jeśli T za małe, to zbiór roboczy nie obejmie całej lokalności procesu
Jeśli T za duże, to zbiór roboczy obejmie kilka lokalności
Jeśli $T = \infty$, to zbiór roboczy obejmie wszystkie strony

Zasada pola roboczego a równoważenie obciążenia

Jeśli są jeszcze wolne ramki $\Rightarrow$ można zainicjować nowy proces

Jeśli $\sum WS_i$ > liczba dostępnych ramek $\Rightarrow$ migotanie $\Rightarrow$ wstrzymanie jednego z procesów

Jak aproksymować zawartość pola roboczego:
zegar + bit odniesienia
Przykład: T = 10,000
- zegar generuje przerwanie co 5,000 jednostek czasu
- na każdą stronę przeznacza się w pamięci dwa bity
- z nadejściem przerwania kopiuje się wszystkie bity odniesienia, a następnie zeruje
- jeśli jeden z bitów w pamięci =1 $\Rightarrow$ strona należy do WS

Inne zagadnienia:

1. Sprowadzanie wstępne; OBL (ang. *one block lookahead*)
2. Wybór rozmiaru strony: fragmentacja, rozmiar tablicy stron, narzut na wejście-wyjście, lokalność
3. Struktura programu: przykład dostępu do macierzy kwadratowej wierszami vs. kolumnami
   Restrukturyzacja programu: zwiększenie lokalności odwołań

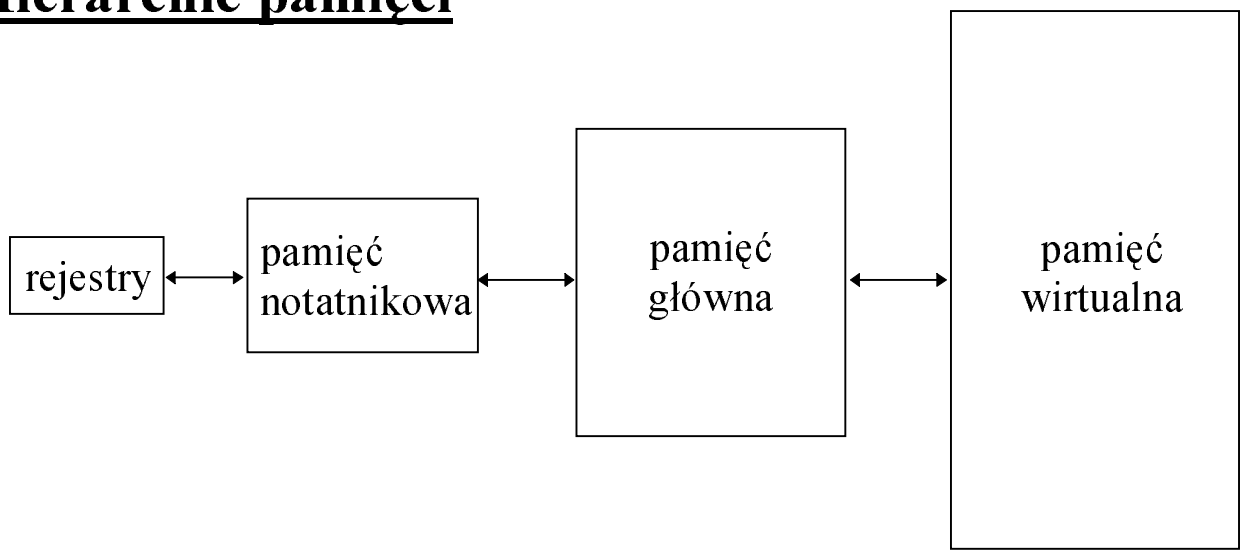
4.Blokowanie stron na czas operacji wejścia-wyjścia

Segmentacja na żądanie

Używana w sytuacji, gdy brak sprzętu do implementacji stronicowania na żądanie

- *Problem umieszczania* jest poważniejszy od problemu wymiany
- OS/2 przydziela pamięć segmentami, które są opisane za pomocą *deskryptorów segmentów*
- Deskryptor segmentu zawiera *bit poprawności* wskazujący, czy segment znajduje się w pamięci (gdy nie, to pojawia się błąd braku segmentu)

Hierarchie pamięci



b. szybka 128 - 4K bajtów	b. szybka 32K - 4M bajtów	szybka 4M - 512M bajtów	wolna 40M - 8G bajtów

Pamięć notatnikowa (ang. *cache memory*)

- Czas dostępu: zbliżony do cyklu procesora
- *Współczynnik trafień* - miara efektywności pamięci notatnikowej (zwykle > 0.9, gdyż programy mają własność lokalności)
- Sposoby organizacji, sposoby zapisu, inicjowanie